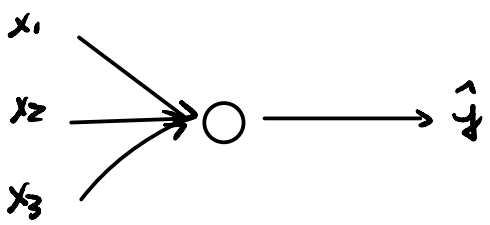
Normalizing activations in a network.

Remember: Normalizing inputs can speed up learning.



$$\mu = \frac{1}{m}\sum_i x^{(i)}$$

$$x = x - \mu$$

$$\sigma^2 = \frac{1}{m}\sum_i x^{(i)2}$$

$$x = x/\sigma^2$$

Batch normalization: Normalize $z^{(i)}$, to make it even faster.

Implementation:

Given some intermediate value in NN, $z^{(1)} \dots z^{(m)}$

$$\mu = \frac{1}{m}\sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m}\sum_i (z_i - \mu)^2$$

$$z^{(i)}_{norm} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}}$$

$$\tilde{z}^{(i)} = \gamma z^{(i)}_{norm} + \beta$$

learnable parameters of model.

If $\gamma = \sqrt{\sigma^2 + \varepsilon}$, $\beta = \mu$ $\Rightarrow \tilde{z}^{(i)} = z^{(i)}$

Use $\tilde{z}^{[l](i)}$ instead of $z^{[l](i)}$

Fitting Batch Norm into Neural Network.

$$X \xrightarrow{W^{[1]}, b^{[1]}} z^{[1]} \xrightarrow[\text{Batch Norm (BN)}]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \longrightarrow a^{[1]} = g^{[1]}(\tilde{z}^{[1]}) \longrightarrow \dots$$

Parameters: $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \dots, w^{[L]}, b^{[L]}$

$\rightarrow \beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \dots, \beta^{[L]}, \gamma^{[L]}$

update BN parameters.

$d\beta^{[l]}$, $\beta^{[l]} = \beta^{[l]} - \alpha d\beta^{[l]}$

$\rightarrow$ tf.nn.batch_normalization.

Working with mini-batches.

$$X^{\{1\}} \longrightarrow z^{[1]} \xrightarrow[BN]{\beta^{[1]}, \gamma^{[1]}} \tilde{z}^{[1]} \longrightarrow \dots$$

Params: $w^{[l]}$, ~~$b^{[l]}$~~, $\mu^{[l]}$, $\gamma^{[l]}$ $\rightarrow$

$z^{[l]} = w^{[l]} a^{[l-1]} + ~~b^{[l]}~~$

$z^{[l]} = w^{[l]} a^{[l-1]}$

$z^{[l]}_{norm} \dots$

$\tilde{z}^{[l]} = \gamma^{[l]} z_{norm} + \beta^{[l]}$

Batch normalization has a slight regularization effect.

- Each mini-batch is scaled by mean/var computed on just mini-batch.
- This adds noice to $z^{[l]}$ ($a^{[l]}$), same effect as dropout.

↑ Hidden layer's activations.